ROBOFUN.RO

LECȚIA XI

CURS GRATUIT

ARDUINO ȘI ROBOTICĂ

Muzică cu Arduino

MP3 Player Shield

Shield-ul MP3 Player contine chip-ul VS1053b, capabil sa decodeze stream-uri MP3, OGG Vorbis, AAC, WMA, MIDI, si de asemenea contine si un slot de card microSD pentru incarcarea fisierelor audio. Shield-ul mai contine si un conector pentru casti sau boxe audio, astfel ca in final, ceea ce obtii este un player MP3 complet.

Ce ai tu de facut este sa citesti informatia stocata pe SD card si sa o trimiti catre chip-ul MP3, atunci cand acesta o solicitia. Suna complicat in teorie, dar din fericire exista deja mai multe librarii care fac asta in locul tau. Cea mai interesanta este libraria disponibila ca download la adresa http://www.robofun.ro/mp3_player_shield (link-ul "Librarie pentru Arduino"). Fisierul .zip pe care il descarci contine atat libraria MP3, cat si libraria pentru SD card. Va trebui sa le copiezi pe ambele in directorul "libraries" din mediul tau de dezvoltare Arduino. Codul sursa este relativ simplu de inteles, toata partea complexa este ascunsa de librarie.

```
#include <SPI.h>
#include <SdFat.h>
#include <SdFatUtil.h>
#include <SFEMP3Shield.h>

SFEMP3Shield MP3player;

byte temp;
byte result;

char title[30];
char artist[30];
char album[30];


void setup() {

  Serial.begin(115200);

  result = MP3player.begin();
  if(result != 0) {
    Serial.print("Error code: ");
    Serial.print(result);
    Serial.println(" when trying to start MP3 player");
    }

  Serial.println("STARTED");

}
```

```
void loop() {
      result = MP3player.playMP3("melodie1.mp3");
      delay(3000);
      if (MP3player.isPlaying()){
        MP3player.stopTrack();
      }

}
```

Avem posibilitatea de a porni redarea unui fisier mp3 la alegere, putem verifica daca s-a terminat intre timp redarea audio (daca s-a terminat fisierul) sau putem opri redarea intr-un moment ales de noi. Spre exemplu, in codul sursa de mai sus, pornim redarea pentru fisierul "melodie1.mp3", si daca dupa 3 secunda fisierul inca nu s-a terminat, atunci il oprim noi fortat.

Libraria mai permite sa facem fastForward pe un fisier pana la o anume pozitie folosind "MP3player.skipTo(<pozitieInSecunde>);" si putem determina pozitia curenta ("MP3player.currentPosition();"). Spre exemplu, "MP3player.skipTo(30000);" va derula mp3-ul curent pana la secunda 30, iar "int pozitie = MP3player.currentPosition();" va incarca in variabila "pozitie" timpul in milisecunde de la inceperea redarii.

Pinii ocupati de acest shield sunt 2, 3, 4, 6, 7, 8, 9, 11, 12, 13 (aproape toti, pentru Arduino UNO). Daca ai nevoie de mai multi pini, poti folosi Arduino Mega in locul lui Arduino UNO sau poti schimba MP3 Player Shield-ul cu un MP3 Trigger (care are nevoie de mult mai putini pini).

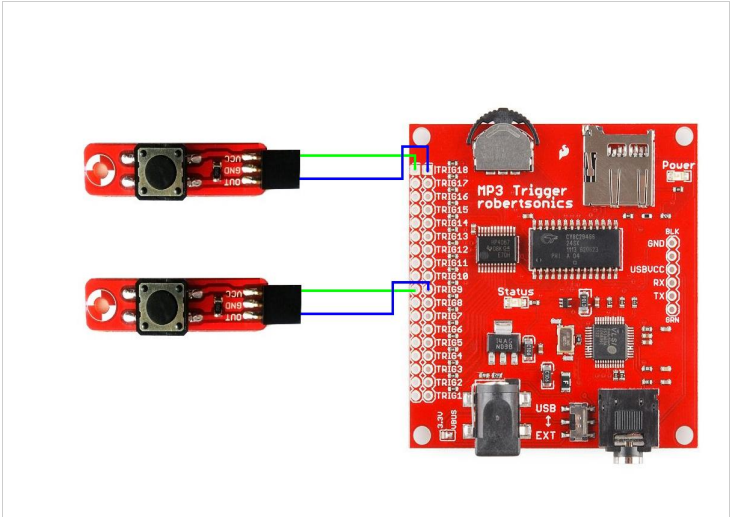
MP3 Trigger

MP3 Trigger este varianta mult imbunatatita a lui shield-ului MP3 Player prezentat in sectiunea anterioara. Pe langa chip-ul capabil sa redea MP3-uri, SD Card-ul deja prezent pe placa, MP3 Trigger-ul are in plus si un microcontroller pre-programat. Astfel, numarul de pini necesari pentru interfatarea cu Arduino scade drastic (doar doi pini sunt necesari) si in plus, MP3 Trigger-ul poate functiona chiar si standalone, fara Arduino. Dat fiind ca este cel mai simplu, sa incepem cu acest mod de functionare.

MP3 Trigger-ul ofera 18 pini, fiecare dintre ei declansand redarea melodiei al carei nume incepe cu numarul asociat pinului. Astfel, spre exemplu, atunci cand pinul 3 este conectat la pinul GND, este redata melodia al carei nume incepe cu "003" (un exemplu de nume valid este "003 Avion cu Motor.mp3".). O schema de conectare folosind butoane brick este mai jos. Am pus in schema doar doua butoane, pentru exemplificare. Evident ca tu poti conecta cate ai nevoie, maxim 18 butoane.

Buton 1 VCC	MP3 Trigger TRIG18 (pinul din interior)
Buton 1 OUT	MP3 Trigger TRIG18 (pinul din exterior)
Buton 2 VCC	MP3 Trigger TRIG9 (pinul din interior)
Buton 2 OUT	MP3 Trigger TRIG9 (pinul din exterior)

Pinii din interior marcati "TRIGNN" sunt conectati la microcontroller-ul placii, si atunci cand unul dintre acesti pini este conectat la GND, incepe redarea melodiei corespunzatoare de pe SD Card. In mod convenabil, toti pinii din sirul exterior sunt conectati deja la pinul GND. Tot ce ai tu de facut atunci cand vrei sa porneasca redarea unei melodii, este sa faci contact intre pinul din interior si pinul din exterior. Butonul brick, in conectarea de mai sus, conecteaza pinul din exterior (GND) cu pinul din interior (TRIGNN) atunci cand este apasat. Acest mod de utilizare este foarte util pentru situatiile cand ai nevoie sa pornesti redarea unei melodii MP3 ca raspuns la un stimul extern, situatie in care nu mai ai nevoie de Arduino. Pentru situatiile mai complexe, MP3 Trigger-ul ofera un API elaborat, accesibil peste un protocol serial TTL, folosind Arduino. Schema de conectare este cea de mai jos.



Arduino 5V	MP3 Trigger USBVCC
Arduino GND	MP3 Trigger GND
Arduino Digital 7	MP3 Trigger TX
Arduino Digital 8	MP3 Trigger RX

Mai departe, tot ce ai de facut este sa folosesti libraria SoftwareSerial pe pinii 7 si 8 si sa-i trimiti placii MP3 Trigger comenzi peste conexiunea seriala. Cele mai des folosite comenzi sunt disponibile mai jos.

Comanda: Start/Stop

Numar de bytes: 1

Byte de comanda: ‘O’

Functionare: Daca exista o melodie care este redata la momentul primirii comenzii, se opreste redarea. Altfel, incepe redarea.

Comanda: Inainte

Numar de bytes: 1

Byte de comanda: ‘F’

Functionare : Urmatoarea melodie MP3 este redata.

Comanda: Inapoi

Numar de bytes: 1

Byte de comanda: ‘R’

Functionare: Melodia precedenta MP3 este redata.

Comanda: Trigger (binary)

Numar de bytes: 2

Byte de comanda: ‘t’

Byte de date: n = 1 to 255

Functionare: Daca exista, melodia cu numele “NNNxxxx.MP3” este redata, unde NNN is echivalentul ASCII al bitului de comanda.

Comanda: Play (binary)

Numar de bytes: 2

Byte de comanda: ‘p’

Byte de date: n = 0 to 255

Functionare: Daca exista, melodia numarul n va fi redata.

Comanda: Set Volume (binary)

Numar de bytes: 2

Byte de comanda: ‘v’

Byte de date: n = 0 to 255

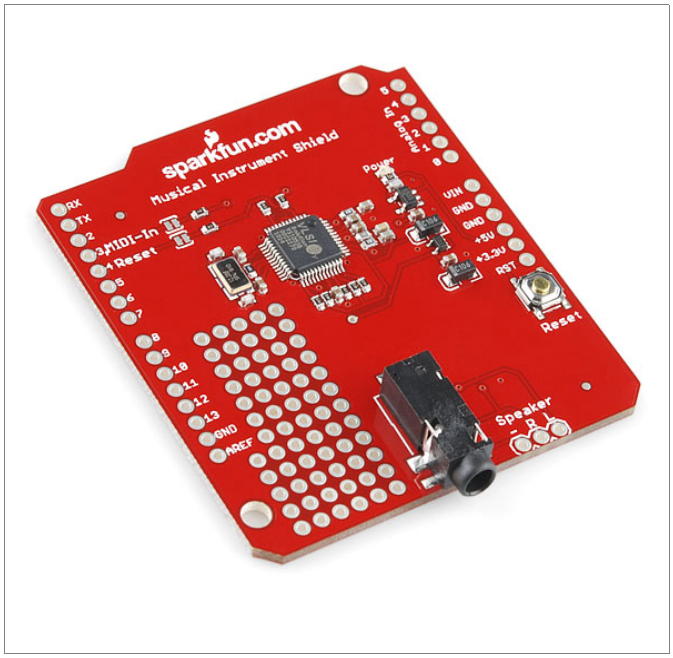
Comments: Volumul va fi setat la valoarea n. Volumul maxim se obtine cu valoarea "0", iar volumul minim in jur de 100.

Toata gama de comenzi suportata este prezentata pe larg in manual de utilizare al placii, disponibil la adresa http://www.robofun.ro/mp3_trigger_v2

Music Instrument Shield

Music Instrument Shield este capabil sa genereze note muzicale conform standardului MIDI. Shield-ul comunica serial cu Arduino, pe pinii digitali 2 si 3, pinul 4 digital functionand drept pin de reset. Controlul shield-ului se face trimitand comenzi pe interfata seriala formata din pinii 2 si 3. Comenzile seriale de control nu sunt foarte simple (lucru care se datoreaza protocolului MIDI, care, in opinia mea, este destul de incurcat).

Shield-ul este capabil sa redea 248 de instrumente, de la cele mai comune (pian, harpa, acordeon, chitara), pana la unele destul de ciudate (latrat de caine, impuscatura, elicopter). Instrumentele sunt organizate in 3 "bank-uri".



Cea mai simpla abordare pentru a intelege modul cum se foloseste acest shield este sa urmarim mai jos un exemplu de program scris de Nathan Seidle de la Sparkfun, program care genereaza cate zece note pentru fiecare instrument inclus.

```
/*
 2-12-2011
 Spark Fun Electronics 2011
 Nathan Seidle

 This code is public domain but you buy me a beer if you use this and we
meet someday (Beerware license).

 This code works with the VS1053 Breakout Board and controls the VS1053 in
what is called Real Time MIDI mode.
 To get the VS1053 into RT MIDI mode, power up the VS1053 breakout board
with GPIO0 tied low, GPIO1 tied high.

 I use the NewSoftSerial library to send out the MIDI serial at 31250bps.
This allows me to print regular messages
 for debugging to the terminal window. This helped me out a ton.

 5V : VS1053 VCC
 GND : VS1053 GND
 D3 (SoftSerial TX) : VS1053 RX
 D4 : VS1053 RESET

 Attach a headphone breakout board to the VS1053:
 VS1053 LEFT : TSH
```

```
 VS1053 RIGHT : RSH
 VS1053 GBUF : GND

 When in the drum bank (0x78), there are not different instruments, only
different notes.
 To play the different sounds, select an instrument # like 5, then play
notes 27 to 87.

 To play "Sticks" (31):
 talkMIDI(0xB0, 0, 0x78); //Bank select: drums
 talkMIDI(0xC0, 5, 0); //Set instrument number
 //Play note on channel 1 (0x90), some note value (note), middle velocity
(60):
 noteOn(0, 31, 60);

 */
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); //Soft TX on 3, we don't use RX in this code

byte note = 0; //The MIDI note value to be played
byte resetMIDI = 4; //Tied to VS1053 Reset line
byte ledPin = 13; //MIDI traffic inidicator
int instrument = 0;

void setup() {
  Serial.begin(57600);

  //Setup soft serial for MIDI control
  mySerial.begin(31250);

  //Reset the VS1053
  pinMode(resetMIDI, OUTPUT);
  digitalWrite(resetMIDI, LOW);
  delay(100);
  digitalWrite(resetMIDI, HIGH);
  delay(100);
}

void loop() {

  talkMIDI(0xB0, 0x07, 120); //0xB0 is channel message, set channel volume
to near max (127)



/*
  //Demo Basic MIDI instruments, GM1
  //================================================================
  Serial.println("Basic Instruments");
  talkMIDI(0xB0, 0, 0x00); //Default bank GM1

  //Change to different instrument
  for(instrument = 0 ; instrument < 127 ; instrument++) {

    Serial.print(" Instrument: ");
    Serial.println(instrument, DEC);

    talkMIDI(0xC0, instrument, 0); //Set instrument number. 0xC0 is a 1
```

```
data byte command

    //Play notes from F#-0 (30) to F#-5 (90):
    for (note = 30 ; note < 40 ; note++) {
      Serial.print("N:");
      Serial.println(note, DEC);

      //Note on channel 1 (0x90), some note value (note), middle velocity
(0x45):
      noteOn(0, note, 60);
      delay(50);

      //Turn off the note with a given off/release velocity
      noteOff(0, note, 60);
      delay(50);
    }

    delay(100); //Delay between instruments
  }
  //=================================================================
*/
```

```
  //Demo GM2 / Fancy sounds
  //=================================================================
  Serial.println("Demo Fancy Sounds");
  talkMIDI(0xB0, 0, 0x78); //Bank select drums

  //For this bank 0x78, the instrument does not matter, only the note
  for(instrument = 30 ; instrument < 31 ; instrument++) {

    Serial.print(" Instrument: ");
    Serial.println(instrument, DEC);

    talkMIDI(0xC0, instrument, 0); //Set instrument number. 0xC0 is a 1
data byte command

    //Play fancy sounds from 'High Q' to 'Open Surdo [EXC 6]'
    for (note = 27 ; note < 87 ; note++) {
      Serial.print("N:");
      Serial.println(note, DEC);

      //Note on channel 1 (0x90), some note value (note), middle velocity
(0x45):
      noteOn(0, note, 60);
      delay(50);

      //Turn off the note with a given off/release velocity
      noteOff(0, note, 60);
      delay(50);
    }

    delay(100); //Delay between instruments
```

```
  }



/*
  //Demo Melodic
  //================================================================
  Serial.println("Demo Melodic? Sounds");
  talkMIDI(0xB0, 0, 0x79); //Bank select Melodic
  //These don't sound different from the main bank to me

  //Change to different instrument
  for(instrument = 27 ; instrument < 87 ; instrument++) {

    Serial.print(" Instrument: ");
    Serial.println(instrument, DEC);

    talkMIDI(0xC0, instrument, 0); //Set instrument number. 0xC0 is a 1
data byte command

    //Play notes from F#-0 (30) to F#-5 (90):
    for (note = 30 ; note < 40 ; note++) {
      Serial.print("N:");
      Serial.println(note, DEC);

      //Note on channel 1 (0x90), some note value (note), middle velocity
(0x45):
      noteOn(0, note, 60);
      delay(50);

      //Turn off the note with a given off/release velocity
      noteOff(0, note, 60);
      delay(50);
    }

    delay(100); //Delay between instruments
  }
*/
}


//Send a MIDI note-on message.  Like pressing a piano key
//channel ranges from 0-15
void noteOn(byte channel, byte note, byte attack_velocity) {
  talkMIDI( (0x90 | channel), note, attack_velocity);
}

//Send a MIDI note-off message.  Like releasing a piano key
void noteOff(byte channel, byte note, byte release_velocity) {
  talkMIDI( (0x80 | channel), note, release_velocity);
}

//Plays a MIDI note. Doesn't check to see that cmd is greater than 127, or
// that data values are less than 127
void talkMIDI(byte cmd, byte data1, byte data2) {
  digitalWrite(ledPin, HIGH);
```

```
  mySerial.write(cmd);
  mySerial.write(data1);

  //Some commands only have one data byte. All cmds less than 0xBn have 2
//data bytes
  //(sort of: http://253.ccarh.org/handout/midiprotocol/)
  if( (cmd & 0xF0) <= 0xB0)
    mySerial.write(data2);

  digitalWrite(ledPin, LOW);
}
```

Primul lucru la care merita sa ne oprim este declaratia conexiunii seriale de tip sofware pe pinii 2 si 3 : "*SoftwareSerial mySerial(2, 3);*". Vom folosi aceasta conexiune seriala pentru a trimite date catre shield. Rata de transfer pentru aceasta conexiune seriala (rata presetata in shield) este de 31250 bps (motiv pentru care avem in setup linia *mySerial.begin(31250)*). Tot in setup dam o comanda de reset shield-ului (folosim pinul 4 digital, care este conectat la pinul de reset al shield-ului, pin pe care il coboram in LOW si apoi il ridicam in HIGH dupa 100 de milisecunde).

Prima instructiune din *loop* este *talkMIDI(0xB0, 0x07, 120)*, care seteaza volumul shield-ului la valoarea 120 (sunt posibile 127 de valori, intre 0 si 127).

Codul este apoi structurat in 3 sectiuni ("*Demo Basic MIDI instruments*", "*Demo GM2 / Fancy sounds*" si "*Demo Melodic*"). La inceputul fiecarei sectiuni avem cate o comanda care seteaza "bank-ul" (una dintre cele trei categorii de instrumente). Astfel, la inceputul primei sectiune avem comanda "*talkMIDI(0xB0, 0, 0x00)*", care seteaza bank-ul de instrumente MIDI clasice, apoi la inceputul celei de-a doua sectiuni avem "*talkMIDI(0xB0, 0, 0x78)*" care seteaza bank-ul cu instrumente de percutie, iar apoi in cea de-a treia sectiune avem "*talkMIDI(0xB0, 0, 0x79)*", care seteaza bank-ul de instrumente melodice.

In prima sectiune avem doua instructiuni *for*. Prima instructiune *for* cicleaza printre cele 128 de instrumente disponibile in acest bank, iar cel de-al doilea *for* genereaza 10 note muzicale pentru fiecare instrument. Comanda interesanta in aceasta sectiune este "*talkMIDI(0xC0, instrument, 0)*", care seteaza instrumentul care va reda notele muzicale. Pentru bank-ul 0X00 (setat anterior prin comanda *talkMIDI(0xB0, 0, 0x00)*, avem 128 de instrumente posibile, intre 0 si 128. Cel de-al doilea *for* genereaza zece note pentru fiecare instrument. Redarea unui note muzicale este pornita cu "*noteOn(0, note, 60)*" si este oprita cu "*noteOff(0, note, 60)*".

Cea de-a doua sectiune ("*Demo GM2 / Fancy sounds*") este ceva mai speciala decat celelalte doua. Acest lucru se intampla din cauza protocolului MIDI, care specifica faptul ca atunci cand este selectat bank-ul cu instrumente de percutie, comanda care seteaza instrumentul nu mai are nici un efect, ci fiecare instrument este de fapt o nota. Alt mod de a spune acelasi lucru este ca in bank-ul cu instrumente de percutie este un singur instrument cu foarte

multe note, iar fiecare nota de fapt reda un alt instrument. Prima comanda din aceasta sectiune selecteaza bank-ul cu instrumente de percutie ("*talkMIDI(0xB0, 0, 0x78)*"). Urmatoarea instructiune *for* va face exact un singur ciclu (pentru ca asa cum am spus mai sus, instrumentul nu conteaza in cazul bank-ului cu instrumente de percutie. A doua instructiune *for* din aceasta sectiune cicleaza intre 27 si 87, cele 60 de instrumente disponibile pentru bank-ul de percutie. Instructiunea "*noteOn(0, note, 60)*" genereaza o nota pe cate unul dintre cele 60 de instrumente.

Cea de-a treia sectiune este perfect similara cu prima sectiune, doar ca in bank-ul cu instrumente melodice instrumentele sunt intre 27 si 87.

Sa analizam acum mai atent functia *noteOn*, care genereaza o anumita nota muzicala. Functia *noteOn* primeste trei parametri. Cel de-al doilea parametru este nota muzicala, iar ultimul parametru reprezinta cat de "puternic" este generata nota (daca ne gandim ca folosim un pian, atunci ultimul parametru reprezinta cat de puternic apasam pe clapa pianului).

In rezumat :

–3 bank-uri cu instrumente

–*talkMIDI(0xB0, 0, 0x00)* – selecteaza bank-ul cu instrumente clasice

–*talkMIDI(0xB0, 0, 0x78)* – selecteaza bank-ul cu instrumente de percutie

–talkMIDI(0xB0, 0, 0x79) – selecteaza bank-ul cu instrumente melodice

–*noteOn(0, note, 120)* – reda nota muzicala *note;* pentru bank-ul 0x78 valoarea lui *note* selecteaza si instrumentul muzical (pentru bank-ul 0x78, valorile posibile pentru *note* sunt intre 27 si 87).

–*noteOff(0, note, 120)* – opreste redarea notei muzicala *note*;

–*talkMIDI(0xB0, 0x07, volum)* – stabileste volumul instrumentului; valoarea pentru *volum* este intre 0 si 127.

–*talkMIDI(0xC0, instrument, 0)* – stabileste instrumentul care va reda notele muzicale; acest lucru este valabil doar pentru bank-urile 0X00 si 0X79; pentru bank-ul 0X78, aceasta comanda nu are nici un efect.

Aceasta a fost lectia 11. In final, as vrea sa te rog sa ne oferi feedback asupra acestei lectii, pentru a ne permite sa le facem mai bune pe urmatoarele.

Este vorba despre un sondaj cu 4 intrebari (oricare este optionala), pe care il poti accesa dand click aici.

Sau ne poti contacta direct prin email la contact@robofun.ro .

Iti multumim,

Echipa Robofun.RO